Mathematical
Structures and Modeling
2018. N. X(XX). PP. 1–??

UDC 000.000

WHY DEEP LEARNING METHODS USE KL DIVERGENCE INSTEAD OF LEAST SQUARES: A POSSIBLE PEDAGOGICAL EXPLANATION

Olga Kosheleva
Ph.D. (Phys.-Math.), Associate Professor, e-mail: olgak@utep.edu
Vladik Kreinovich
Ph.D. (Phys.-Math.), Professor, e-mail: vladik@utep.edu

University of Texas at El Paso, El Paso, Texas 79968, USA

Abstract. In most applications of data processing, we select the parameters that minimize the mean square approximation error. The same Least Squares approach has been used in the traditional neural networks. However, for deep learning, it turns out that an alternative idea works better – namely, minimizing the Kullback-Leibler (KL) divergence. The use of KL divergence is justified if we predict probabilities, but the use of this divergence has been successful in other situations as well. In this paper, we provide a possible explanation for this empirical success. Namely, the Least Square approach is optimal when the approximation error is normally distributed – and can lead to wrong results when the actual distribution is different from normal. The need to have a robust criterion, i.e., a criterion that does not depend on the corresponding distribution, naturally leads to the KL divergence.

Keywords: Deep learning, Kullback-Leibler divergence.

1. Formulation of the Problem

Machine learning: reminder. The main problem of machine learning is:

- given input-output patterns $\left(x^{(k)}, y^{(k)}\right)$,
- to come up with a function $f(x)$ for which $f\left(x^{(k)}\right) \approx y^{(k)}$.

This function can then be used to predict the output y for other inputs x.

In each model of machine learning:

- we have a function $f(x, c)$ depending on some parameters $c = (c_1, c_2, \ldots)$, and
- we need to find the values of these parameters for which the resulting values $z^{(k)} \stackrel{\text{def}}{=} f\left(x^{(k)}, c\right)$ are approximately equal to the given value $y^{(k)}$:

$$z^{(k)} \approx y^{(k)}.$$

How to describe this approximate equality: traditional approach. Traditionally, in machine learning, the Least Squares approach was used to describe the desired approximate equality of:

- the result $z^{(k)}$ of applying the model $f(x, c)$ to the input $x^{(k)}$ and
- the given outputs $y^{(k)}$;

see, e.g., [1]. Specifically, most traditional methods minimize the sum

$$\sum_{k=1}^{K} \left(z^{(k)} - y^{(k)}\right)^2. \tag{1}$$

Deep learning techniques use KL divergence instead. In deep learning, it turned out that better results are obtained if, instead of the least squares technique (1), we use the Kullback-Leibler (KL) divergence; see, e.g., [2, 3]. Specifically, we re-scale the values $y^{(k)}$ and $z^{(k)}$ so that these values are always between 0 and 1, and then minimize the following objective function:

$$-\sum_{k=1}^{K} \left[y^{(k)} \cdot \log\left(z^{(k)}\right) + \left(1 - y^{(k)}\right) \cdot \log\left(1 - z^{(k)}\right)\right]. \tag{2}$$

Why? At first glance, the least squares is a reasonable criterion, the one most frequently used in statistical data processing; see, e.g., [4]. So why is the alternative approach working better?

For the case when the predicted values $y^{(k)}$ are probabilities, an explanation is given in [2], Section 5.5. However, the criterion (2) is also successfully used in many applications in which the predicted values $y^{(k)}$ are not probabilities. How can we explain this success?

What we do in this paper. In this paper, we extend the existing probability-case explanation to the general case, thus providing a possible explanation of why KL divergence works well.

2. Our Explanation

Why least squares: reminder. In order to explain why KL divergence is efficient, let us first recall why the Least Squares method is often used.

Ideally, the deviations $z^{(k)} - y^{(k)}$ should be all 0s, but in reality, we can only attain an approximate equality. In different situations, we get different values of these deviations. It is therefore reasonable to view these deviations as random variables.

In practice, many random variables are normally distributed; see, e.g., [**?**]. It is therefore reasonable to assume that the deviations $z^{(k)} - y^{(k)}$ are normally distributed, with 0 means and some standard deviation σ. The corresponding probability density function is thus equal to

$$\rho\left(z^{(k)} - y^{(k)}\right) = \frac{1}{\sqrt{2\pi}\cdot\sigma}\cdot\exp\left(-\frac{\left(z^{(k)} - y^{(k)}\right)^2}{2\sigma^2}\right). \tag{3}$$

Since we do not have any reason to believe that different deviations are positively or negatively correlated, it is reasonable to assume that different deviations are independent. In this case, for each tuples c, the probability (density) is equal to the product of the corresponding probabilities (3), i.e., equal to

$$\rho(c) = \prod_{k=1}^{K}\left[\frac{1}{\sqrt{2\pi}\cdot\sigma}\cdot\exp\left(-\frac{\left(z^{(k)} - y^{(k)}\right)^2}{2\sigma^2}\right)\right], \tag{4}$$

where $z^{(k)} = f(x^{(k)}, c)$. It is reasonable to select the tuple c which is the most probable, i.e., for which the expression (4) is the largest possible; this natural idea is known as the *Maximum Likelihood approach*.

Maximizing the expression (4) is equivalent to minimizing its negative logarithm

$$-\ln(\rho(c)) = \text{const} + \sum_{k=1}^{K}\frac{\left(z^{(k)} - y^{(k)}\right)^2}{2\sigma^2},$$

and this minimization is equivalent to minimizing the Least Squares expression (1).

Need to go beyond the Least Squares. While many practical probability distributions are normal, there are also many cases when the probability distribution is different form normal; see, e.g., [4]. In such cases, the Least Squares method is not optimal – and it can be very far form optimal. For example, if we have a distribution with heavy tails, for which the probability of large deviations is high, the Least Squares method often leads to erroneous estimates.

This can be illustrated on the simple example when the model $f(x, c) = c_1$ is simply an unknown constant c_1. In this case, if we minimize the sum $\sum\limits_{k=1}^{K}\left(x^{(k)} - c_1\right)^2$ – by differentiating by c_1 and equating the derivative to 0 – we get the estimate $\hat{c}_1 = \frac{1}{K}\cdot\sum\limits_{k=1}^{K} y^{(k)}$. If the actual value of c_1 is, e.g., 10, and we get $K = 100$ values close to 10, then the arithmetic average is indeed close to 10. But if one of the estimates is an outlier, e.g., $x^{(1)} = 10^6$, then the arithmetic average is close to 10,000 – way beyond the actual value 10.

To take this non-normality into account, we need to replace the Least Squares approach with a one which is *robust*, in the sense that it does not depend on the probability distribution of the divergence.

Main idea. In the computer, any real value is represented as 0s and 1s. To transform a real-valued signal into a sequence of 0s and 1s, measuring instruments use analog-to-digital converters. These converters are usually based on comparing the actual value with some threshold values. For example, if we the actual value is between 0 and 1, then, by comparing this value with 0.5, we can tell whether the first bit in the binary expansion is 0 or 1:

- if the actual value is smaller than 0.5, then the first bit is 0, and
- if the actual value is larger than 0.5, then the first bit is 1.

By selecting a second threshold to be 0.25 or 0.75, we can determine the second bit, etc.

The thresholds are not necessarily binary-rational numbers: often, other thresholds are used, and then the resulting number is recovered from the results of the corresponding comparisons.

We want to come up with a probabilistic interpretation; thus, it makes sense to select random thresholds. The simplest possible random number generator generates values uniformly distributed on the interval $[0, 1]$. Such random number generators are included in most programming languages.

Resulting setting. So, to describe each value $y^{(k)}$, let us run this simplest random number generator a large number of time N, and store N results of comparing $y^{(l)}$ with the corresponding random numbers r_i:

- we store 1 if $r_i \le y^{(k)}$, and
- we store 0 if $r_i > y^{(k)}$.

For a random number uniformly distributed on the interval $[0, 1]$, the probability to be in each interval is equal to the width of this interval. In particular, the probability to be smaller than or equal to $y^{(k)}$ – i.e., the probability to be in the interval $\left[0, y^{(k)}\right]$ – is equal to $y^{(k)}$. Thus, for large N, we have:

- approximately $N \cdot y^{(k)}$ 1's and
- approximately $N \cdot \left(1 - y^{(k)}\right)$ 0s.

For each of K patterns, we have N 0-1 records, so overall, we have a long sequence $N \cdot K$ records corresponding to all K patterns. This sequence corresponds to the observations.

Derivation of KL divergence. We want to find the tuple c that best fits the above long sequence of observations.

For each tuple c and for each pattern k, the model $f(x, c)$ returns the value $z^{(k)} = f\left(x^{(k)}, c\right)$. Thus:

- the probability to get 1 when we compare this value with a random value r_i is equal $z^{(k)}$, and

- the probability to get 0 is equal to the remaining probability $1 - z^{(k)}$.

So, for each pattern, we have:

- $N \cdot y^{(k)}$ observations with probability $z^{(k)}$, and
- $N \cdot \left(1 - y^{(k)}\right)$ observations with probability $1 - z^{(k)}$.

Assuming – as before – that all observations are independent, we conclude that the probability of observing the given sequence of 0s and 1s is equal to the product of all these probabilities, i.e., to the value

$$\left(z^{(k)}\right)^{N \cdot y^{(k)}} \cdot \left(1 - z^{(k)}\right)^{N \cdot \left(1 - y^{(k)}\right)}.$$

The overall probability can be obtained by multiplying probabilities corresponding to all K patterns. Thus, we select a tuple c for which the following expression is the largest possible:

$$p \stackrel{\text{def}}{=} \prod_{k=1}^{K} \left[\left(z^{(k)}\right)^{N \cdot y^{(k)}} \cdot \left(1 - z^{(k)}\right)^{N \cdot \left(1 - y^{(k)}\right)}\right].$$

Maximizing this expression p is equivalent to minimizing its negative logarithm

$$-\ln(p) = -\sum_{k=1}^{K} \left[N \cdot y^{(k)} \cdot \ln\left(z^{(k)}\right) + N \cdot \left(1 - y^{(k)}\right) \cdot \ln\left(1 - z^{(k)}\right)\right].$$

This expression is N times larger than the KL divergence (3). Thus, minimizing this expression is indeed equivalent to minimizing the KL divergence.

So, we indeed get the desired explanation for minimizing the KL divergence.

Acknowledgments

This work was supported in part by the National Science Foundation grant HRD-1242122 (Cyber-ShARE Center of Excellence).

References

1. C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer Verlag, New York, 2006.
2. I. Goodfellow, Y. Bengio, and A. Courville, *Deep Leaning*, MIT Press, Cambridge, Massachusesst, 2016.
3. P. Liu, K.-K. R. Choo, L. Wang, and F. Huang, "SVM or deep learning? A comparisive study on remote sensing image classification", *Soft Computing*, 2017, Vol. 21, pp. 7053–7065.
4. D. J. Sheskin, *Handbook of Parametric and Nonparametric Statistical Procedures*, Chapman and Hall/CRC, Boca Raton, Florida, 2011.